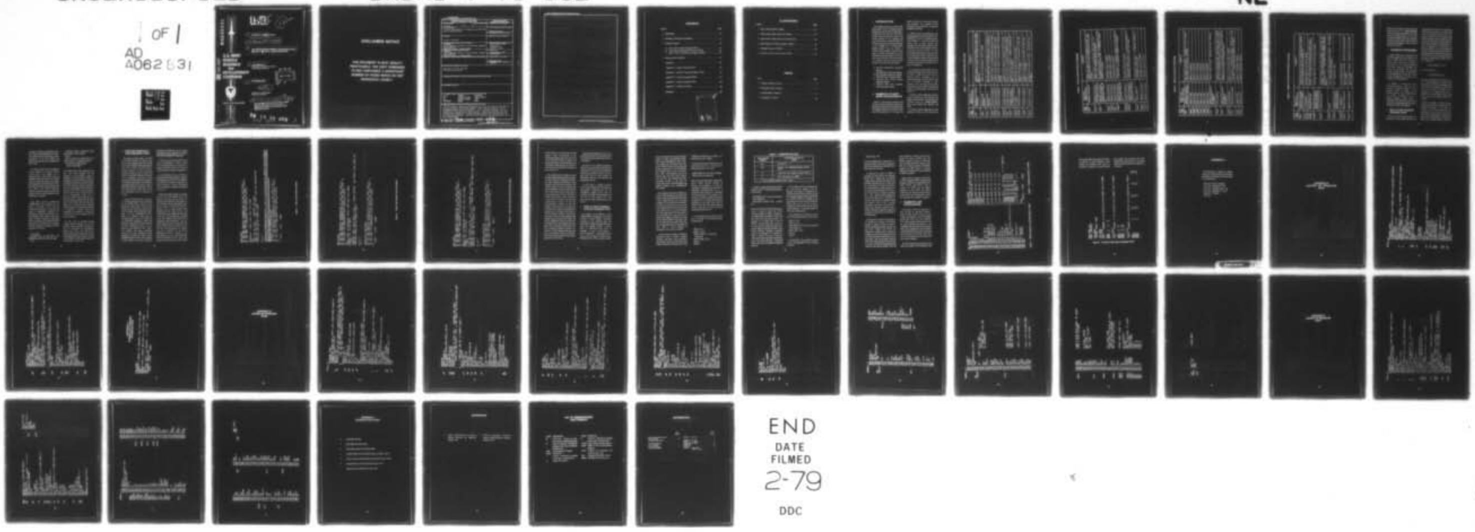
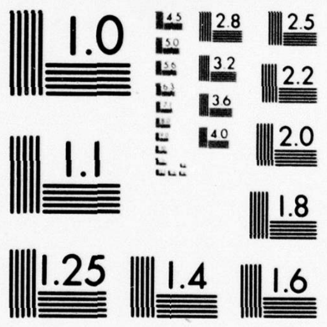
AD-A062 831 ARMY MISSILE RESEARCH AND DEVELOPMENT COMMAND REDSTO--ETC F/G 9/2
SOFTWARE DEVELOPMENT FOR INTERFACING AN HP-21 MX WITH A TEKTRON--ETC(U)
SEP 78 J R MITCHELL

UNCLASSIFIED DRDMI-T-78-102 NL

1 OF 1
AD
A062831



END
DATE
FILMED
2-79
DDC



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF

AD A062831

LEVEL II

12

9 TECHNICAL REPORT T-78-102

14 DRDMI-T-78-102

6

# SOFTWARE DEVELOPMENT FOR INTERFACING AN HP-21 MX WITH A TEKTRONIX 4051

**U.S. ARMY MISSILE RESEARCH AND DEVELOPMENT COMMAND**

DDC FILE COPY

10 Jerrel R. Mitchell
Guidance and Control Directorate
Technology Laboratory

11 29 September 1978

12 45p.

Approved for Public Release;
Distribution Unlimited

16 1W362303A214 17 02

Redstone Arsenal, Alabama 35809

78 12 26 089
393 427

DMI FORM 1000, 1 APR 77

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | | |
|---|---|---|
| 1. REPORT NUMBER<br>T-78-102 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>Software Development for Interfacing an HP-21 MX with a Tektronix 4051 | | 5. TYPE OF REPORT & PERIOD COVERED<br>Technical Report |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>Jerrel R. Mitchell | | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Commander<br>US Army Missile Research and Development Command<br>ATTN: DRDMI-TG<br>Redstone Arsenal, Alabama 35809 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS<br>1W362303A214<br>632303.11.21402 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Commander<br>US Army Missile Research and Development Command<br>ATTN: DRDMI-TI<br>Redstone Arsenal, Alabama 35809 | | 12. REPORT DATE<br>29 September 1978 |
| | | 13. NUMBER OF PAGES<br>55 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | | 15. SECURITY CLASS. (of this report)<br>UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release;
Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

| | | |
|---|---|---|
| RTE-M | FORTRAN | ASSEMBLER |
| HP | CONTROL SYSTEM | BINARY SYSTEM |
| RTE | GRAPHIC SYSTEM | XFER |
| TEKTRONIX | RADAR | TCOM |

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this report, software developments for fully utilizing the marriage between a Tektronix 4051 Graphic System, an HP-21 MX minicomputer, and an HP-9885M flexible disc are presented. First, software necessary for this combination is presented in several tables. Then three programs that were specifically tailored for transferring data between the three devices are presented and discussed. The use of each program is illustrated with one or more examples.

78 12 26 089

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## CONTENTS

## ILLUSTRATIONS

## TABLES

## 1. INTRODUCTION

The standard console for an HP-21 MX minicomputer with an HP-9885M flexible (floppy) disc is an HP-2644/45 data station. In fiscal year 1977 an HP-21 MX with a flexible disc was purchased by the Control Systems Branch of the Guidance and Control Directorate of MIRADCOM. The HP data station was not purchased because of the availability of a Tektronix 4051 Graphic System that can emulate a Tektronix 4012 computer terminal. In this report, software developments for fully utilizing the marriage between the HP-21 MX and the Tektronix 4051 are presented. It is assumed that the reader has some familiarity with the following manuals:

- RTE-M Programmer's Reference Manual
- HP FORTRAN Reference Manual
- RTE Assembler Reference Manual
- RTE-M Editor Reference Manual
- Tektronix 4051 Graphic System Reference Manual
- Tektronix 4051 Data Communications Interface Manual

## 2. SUMMARY OF SOFTWARE DEVELOPMENTS

*Tables 1, 2,* and *3* present a summary of all symbolic and binary programs generated by this author, along with some frequently used HP supplied programs and libraries. Each line of the tables gives a file name, the label(s) of disc(s) on which the file resides, and comments. The comments provide information that should aid in using the programs in the files.

In *Table 3*, there are references to two operating systems, SYSGEN and SYSGN2. SYSGEN is a Type M-I operating system, and SYSGN2 is a Type M-II operating system. Both of these systems were generated for use with the Tektronix 4051, operating in the Terminal Mode[1]. Before entering Terminal Mode, several environmental parameters that control the Communication Interface of the 4051 must be set. Setting these parameters and entering Terminal Mode is easily accomplished by executing, on the 4051, the BASIC program given in Appendix A.

After entering Terminal Mode, either of the above operating systems can be "booted-in" from a floppy disc. There are two different boot-in procedures, one for systems beginning in track 0, sector 2, and one for systems beginning in other locations. (If no track and sector numbers are given on the disc label for an operating system, it can be assumed to start in track 0, sector 2.)

To boot-in systems beginning in track 0, sector 2, bits zero, six, nine, fourteen, and fifteen of the S-register to one (the other bits should be zero), i.e., 1100001001000001. Then, press STORE, IBL, PRESET, and RUN.

To boot-in systems beginning in other locations, set bits six, nine, fourteen, and

**TABLE 1. SYMBOLIC PROGRAMS AND FILES**

| FILE NAME | DISC(S) ON WHICH LOCATED | COMMENTS |
|---|---|---|
| $AB | FORTRAN and Program Development | Assembler routine for returning contents of A and B registers to a FORTRAN Program (See Program Reference Manual Pages 4-5). |
| ECHO | Reduced System Generation | Answer file for generating a TYPE II System. |
| ECHO1 | Reduced System Generation | Echo file resulting from generation of an M-II System can be used as answer file in future generations. Is commented. |
| $HELP | Assembler Development | Assembler routine for satisfying certain calls by % TKHP. |
| $INSUB | Assembler Development | Assembler routine called by subroutine TRDS in TCOM. |
| $IO13 | Assembler Development | Assembler routine called by subroutines $HELP, $INSUB and $OUTSB. |
| $OUTSB | Assembler Development | Assembler routine called by subroutine TRTP in TCOM. |
| SCR1 | FORTRAN Development | Scratch file used in compilation of FORTRAN Programs. |
| SNAP1 | FORTRAN Assembler and Reduced Generation | Snap file for Type I Systems. Needed in relocating programs for Type I Systems. |
| SNAP2 | M-II System | Same as SNAP1 except for Type II System. |
| $TCOM | FORTRAN Development | FORTRAN main program and subroutines of TCOM (See Section 3. b. for usage). |
| $TKHP | FORTRAN Development | FORTRAN main program and subroutine of TKHP (See Section 3. c. for usage.). |
| $XFER | FORTRAN Development | FORTRAN source program of XFER. (See Section 3.A, for usage.) |

TABLE 2. RELOCATABLE BINARY PROGRAMS

| FILE NAME | DISC(S) ON WHICH LOCATED | COMMENTS |
|---|---|---|
| %CAT | Assembler Development | Program that reads up to 128 characters from 4051. The read is terminated by D and the characters are returned to the 4051. |
| %FF4.N | FORTRAN Development and Program Development | FORTRAN library supplied by HP. First library to be searched when relocating FORTRAN programs. |
| %FMPF | FORTRAN Development Program Development and All Generation (Reduced, too) | File management package. Should be searched when any file management routine is being used. In FORTRAN, second library to be searched. |
| %HELP | Assembler Development | Relocatable version of $HELP2. |
| %INSUB | Assembler Development | Relocatable version of $INSUB. |
| %I013 | Assembler Development | Relocatable version of $1013. |
| %MSYLB | FORTRAN Development, Program Development and All Generation (Reduced, too) | System library. In FORTRAN relocation is second library to be searched if file management package is not needed. |
| %OUTSB | Assembler Development | Relocatable version of $OUTSB. |
| %RLIB1 | FORTRAN Development, Program Development and All Generation (Reduced, too) | Floating Point Library, Part 1. In FORTRAN relocation, is third library to be searched if FMP is not needed. |
| %RLIB2 | (Same as %RLIB1) | Floating Point Library, Part 2. In FORTRAN relocation is last library to be searched. |
| %TKHP | FORTRAN Development | Relocatable version of $TKHP2 |
| %TCOM | FORTRAN Development | Relocatable version of $TCOM. |

TABLE 3. ABSOLUTE BINARY PROGRAMS

| FILE NAME | DISC(S) ON WHICH LOCATED | COMMENTS |
|---|---|---|
| ASM | Assembler Development and Program Development | Main Program of Assembler. Relocated for Type I Systems. Automatically loads segments as needed. |
| ASMB1 | (Same as ASM) | Segment of Assembler |
| ASMB2 | (Same as ASM) | Segment of Assembler |
| ASMB3 | (Same as ASM) | Segment of Assembler |
| ASMB4 | (Same as ASM) | Segment of Assembler |
| ASMBD | (Same as ASM) | Segment of Assembler |
| ASMBX | (Same as ASM) | Segment of Assembler |
| CAT | FORTRAN Development and Assembler Development | Absolute version of %CAT. Relocated for Type I Systems. |
| DSKET | First Generation | Program for formatting discs. Relocated for Type I Systems. (See OVR33 Programming Manual for instructions on usage.) |
| EDIT | FORTRAN Development and Assembler Development | Editor to be run under control of Type I Systems located on FORTRAN and Assembler Development Discs. |
| EDIT2 | M-II Development | Editor to be run under Type II System. |
| FTN | FORTRAN Development | Main Program of HP FORTRAN Compiler. Relocated for Type I Systems. Automatically loads segments as needed. |
| FTN1 | (Same as FTN) | Segment of FORTRAN Compiler. |
| FTN2 | (Same as FTN) | Segment of FORTRAN Compiler. |

TABLE 3. ABSOLUTE BINARY PROGRAMS (Continued)

| FILE NAME | DISC(S) ON WHICH LOCATED | COMMENTS |
|---|---|---|
| RELF | FORTRAN Development and M-II Development | Command file that can be used to place relocation commands which the relocating loader can transfer to. |
| RTLD2 | M-II Development | Relocating loader for the Type II System. |
| RTLD2 RTMGN | First Generation and Reduced Generation | Relocated system generation program for use with Type I systems. Can be used to generate other systems with 4051. |
| RTMLD | FORTRAN Development, Assembler Development and Reduced Generation | Relocating loader for use with the Type I Systems. |
| SGPRP | Program Development | Segmented Program Preparation Program. Runs under Type I Systems. After relocating segmented programs, this program should be run. (See Pages 7-33 of Program Reference Manual.) |
| SYSCPY | SYSCPY | Type I System with single executable program. (See section 3. A. of this report for details). |
| SYSGEN | FORTRAN Development, Assembler Development and Reduced Generation | Type I Operating Systems. |
| SYSGN2 | M-II Development | Type II Operating Systems. |
| TCOM | FORTRAN Development | Absolute version of $TCDM. Relocated for execution under Type I Systems. |
| TKHP | FORTRAN Development | Absolute version of $TKHP. Relocated for execution under Type I System. |

fifteen of the S-register to one (the other bits should be zero), i.e., 1100001001000000. Look up the octal equivalents of the track and sector numbers (see p. F-2 of the RTE-M System Generation Reference Manual). Add the track and sector octal equivalents and store in the B-register. Then, press IBL, PRESET, and RUN.

## 3. TRANSFER PROGRAMS

A major problem that resulted from the available equipment was the inability to be able to transfer files from one disc to another. Also, there was no means to backup files, i.e., by storing on alternate mass media. These two problems were solved by developing the necessary computer coding to allow for file transfers between different discs and for file transfers between the magnetic tape cassette unit on the 4051 and the flexible discs.

In addition to solving the above-mentioned problems, a goal was set to develop the necessary software to allow programs written in BASIC on the Tektronix 4051 to use the HP-9885M for mass storage. To achieve this goal, a monitor program (written partly in FORTRAN and partly in assembler) was writtern for interfacing BASIC program on the 4051 with files on the 9885M.

### A. DISC TO DISC TRANSFER PROGRAM (XFER)

The disc to disc transfer program was developed to aid in transferring files from one flexible disc to another. The underlying principle of the program is to transfer data from a file on one flexible disc to the memory of the HP-21 MX, mount a second flexible disc and transfer the data from the memory to a file on the second flexible disc. The computer coding to accomplish this is given in Appendix B. This program uses several routines from the File Management Package. All these routines are described in the Programmer's Reference Manual except DCMC [2]. This is the mount/dismount routine and is FORTRAN callable. The call for a dismount is

```
CALL DCMC (1, -lu, 0)
```

and for a mount it is

```
CALL DCMC (0, lu, 0)
```

where lu is the logical unit number of the disc. The last argument is the last track number of the disc: if zero, it defaults to that of the disc. If this argument is omitted, the disc will not mount. Actually, for a dismount the first argument needs only to differ from zero, and the disc number can be used in place of the negative logical unit number.

Because of limited computor memory there is a limit on the size of a file that can be transferred. Because of this limit two versions of the copying programs were developed. The first version can be used to transfer files whose lengths are less than 105 blocks (this is approximate). The absolute binary version of this program is XFER. It is located on both the FORTRAN DEV. DISC and the ASSEMBLER DEV. DISC. *The symbolic and relocatable binary reside,* respectively, in the files $XFER and %XFER and are located on the FORTRAN DEV. DISC.

The second version has been included as the lone program in a minimum TYPE I system. Files with lengths up to approximately 150 blocks can be transferred with this version. The TYPE I system in which this version is an integral part is located on the SYSCPY disc. To run this version, "boot-up" on this disc and type RU, XFER or ON, XFER. When the transfer is complete, another system must be booted-in before another program can be loaded and/or executed.

Using either version is simple and straightforward. But remember, *do not mount or dismount a disc until the programs give permission*. If you do, you might be unplesantly surprised. (For example, you might find a directory on a disc changed.) As an example on the use of XFER, consider transferring a file called TEST on one disc to a file called TESTX on another disc. Assuming that the program has been loaded, the dialogue with the computer is as follows:

**ON, XFER*
DISMOUNT & MOUNT (IF DESIRED) AND GIVE FILE NAME
*TEST*
REMOVE DISK & MOUNT NEW DISK & GIVE FILE NAME
*TESTX*
THE FILE TESTX IS CREATED. IT IS TYPE 4 AND IS -1 BLOCKS LONG.
TRANSFER COMPLETED
XFER: STOP 0000

The italicized parts were supplied by the user. (The actual dialogue with the computer for this example is given in Appendix C.) The user has the freedom of loading and running XFER from one disc, mounting a second disc and transferring the file TEST to memory, and then mounting a third disc and transferring the contents of memory to file TESTX. In the case above, TESTX was not found on the disc; thus, a file was automatically created. The "-1 BLOCKS LONG" means that the exact number of blocks needed by the file will be used. If TESTX had existed, the file space defined for it would have been used and extents would have been added if needed. If it is desired to transfer several files sequentially, the program can simply be rerun (however, do not change discs until given permission).

If errors occur in opening, closing, reading, writing, creating, etc., files, the program will automatically terminate and print the appropriate file management negative error codes. The codes can be interpreted by referring to Section IX of the Programmer's Reference Manual. A typical error is "FMP-6," which usually means a file is not found or a disc is full.

## B. DISC-TAPE COMMUNICATIONS PROGRAM (TCOM)

The purpose of this program is to allow file transfers between a magnetic tape cassette on the Tektronix 4051 and a flexible disc on the *HP-9885M. This program can be* used in lieu of XFER, or it can be used to transfer programs to tape for backup purposes. If used in lieu of XFER, the transfer will be slower because the transfer rate is basically controlled by the communications link between the computer and the 4051 (in particular, each character is transmitted serially as ten bits at a rate of 2400 baud). However, binary files of more than 750 blocks and ASCII files of more than 1500 blocks can be transferred by TCOM.

A listing of TCOM is given in Appendix D. The absolute binary version is located in the file TCOM that is located on the FORTRAN DEV. DISC. TCOM can be loaded and run under the control of the TYPE I systems on the FORTRAN DEV. DISC or on the ASSEMBLER DEV. DISC. The symbolic version and the relocatable binary versions of TCOM and its FORTRAN subroutines, TRDS and TRTP, are located in the files $TCOM and %TCOM, respectively, which are also located on the FORTRAN DEV. DISC. The symbolic and binary relocatable versions of the two assembler routines required by TCOM are located on the ASSEMBLER DEV. DISC. The symbolics of these routines are stored in the files $OUTSB and $INSUB, and the binary relocatables of these programs are stored in %OUTSB and %INSUB, respectively.

In order to use the Tektronix 4051 in the Tape Communications Mode, certain environmental parameters must be set[1]. A listing of a BASIC program for setting the required parameters is given in Appendix A.

As an illustration, the use of TCOM is demonstrated by three examples. Listings of the dialogue with TCOM and the dialogue with the BASIC interpreter of the 4051 are shown in *Figures 1, 2,* and *3.* All lines less than ten characters in length were supplied by the user.

In the first example, the goal is to transfer the file CAT to file number four on the magnetic tape. First, the user initiates the execution of TCOM (the assumption here is that TCOM was previously loaded). Next, the user types +1 and presses return to indicate a disc to tape transfer. After mounting the disc with the file to be transferred, the user supplies the file name (in this case, CAT) along with a carriage return. The computer prints three lines of information and instructions. The user selects a tape file by pressing the shift key and the FIND FILE key; then he types 4. The tape is then positioned to file number four. (It is assumed that the DATA COMMUNICATIONS OVERLAY has been placed over the USER DEFINED KEYS on the 4051.) He then types -1 (no carriage return) and presses the DATA

```
*RU,TCOM
IF YOU ARE SENDING INFORMATION TO THE TAPE TYPE +1
IF YOU ARE SENDING INFORMATION FROM THE TAPE TYPE -1
IF YOU WANT TO TERMINATE THIS SESSION TYPE 0
+1
DISMOUNT & MOUNT AND GIVE FILE NAME
CAT
THE FILE IS TYPE 7   PREPARE THE TAPE
WHEN YOU ARE READY, TYPE -1 AND PRESS THE DATA RECEIVE KEY
TERMINAL CAN BE IN PROMPT MODE!

File 4
-1
c@B@@@@B@@@@@B@@D@@@@@@F@@@njP@Ki@qfoP@EK@aGK@qDK@q1J@@EK@@Kt@qV1P@1Q@@1G
@@njP@Ke@qrr@@Km@qrq@@Ky@q1V@@Kq@q1T@@EK@QDP@@1Z@@fpP@EK@aKs@qEK@aCa@qGK@q
[illegible]@@D@@A@@@AdP@@G@@@@@@@AhPF
TRANSFER TO TAPE COMPLETED
IF YOU ARE SENDING INFORMATION TO THE TAPE TYPE +1
IF YOU ARE SENDING INFORMATION FROM THE TAPE TYPE -1
IF YOU WANT TO TERMINATE THIS SESSION TYPE 0
0
  TCOM  : STOP   0000
```

Figure 1.   Disc to tape transfer example.

```
*RU,TCOM
IF YOU ARE SENDING INFORMATION TO THE TAPE TYPE +1
IF YOU ARE SENDING INFORMATION FROM THE TAPE TYPE -1
IF YOU WANT TO TERMINATE THIS SESSION TYPE 0
-1
MOUNT DISC TO RECEIVE; GIVE TYPE AND FILE NAME
FORMAT IS (I1,1X,3A2)
7 CAT2
PREPARE TAPE: WHEN READY TYPE -1, PRESS RETURN  AND PRESS DATA SEND KEY
TERMINAL MUST BE IN PROMPT MODE

File 4
-1

TRANSFER COMPLETED
IF YOU ARE SENDING INFORMATION TO THE TAPE TYPE +1
IF YOU ARE SENDING INFORMATION FROM THE TAPE TYPE -1
IF YOU WANT TO TERMINATE THIS SESSION TYPE 0
0
 TCOM  : STOP   0000
```

**Figure 2.  Tape to disc transfer when file existed.**

```
*RU,TCOM
IF YOU ARE SENDING INFORMATION TO THE TAPE TYPE +1
IF YOU ARE SENDING INFORMATION FROM THE TAPE TYPE -1
IF YOU WANT TO TERMINATE THIS SESSION TYPE 0
-1
MOUNT DISC TO RECEIVE; GIVE TYPE AND FILE NAME
FORMAT IS (I1,1X,3A2)
7 CAT3
FILE NOT FOUND. FILE, CAT3   IS CREATED AS A TYPE: 7 .
PREPARE TAPE: WHEN READY TYPE -1, PRESS RETURN  AND PRESS DATA SEND KEY
TERMINAL MUST BE IN PROMPT MODE

File 4
-1

TRANSFER COMPLETED
IF YOU ARE SENDING INFORMATION TO THE TAPE TYPE +1
IF YOU ARE SENDING INFORMATION FROM THE TAPE TYPE -1
IF YOU WANT TO TERMINATE THIS SESSION TYPE 0
0
  TCOM   : STOP    0000
```

**Figure 3. Tape to disc transfer when file did not exist.**

RECEIVE key. At this point the transfer begins. Some overprinting of the information going to tape will occur on the screen. When the transfer is complete, the program will return to the initial point, and the transfer of other files can be initiated or execution can be terminated. The latter was done in this case.

The examples shown in *Figures 2* and *3* illustrate transfer from tape files to disc files. It is easily seen that the dialogue is similar. One major difference between disc to tape and tape to disc transfers is that tape to disc transfers require the user to supply a file type APPENDIX F and a file name. When a file is transferred from disc to tape, the user must remember the file type if he plans to transfer the file back to a disc. Type 4 files are assumed to be binary. Since the data communications interface of the 4051 communications interface of the 4051 assumes ASCII data only, the transfer of disc binary files to tape is accomplished by coding each 16 bit binary word into four ASCII characters. (Because of this, binary files require twice as much magnetic tape storage as disc storage.) Thus, when files that are designated as binary are transferred from tape to disc, a decoding process takes place. On the other hand, ASCII files (type 4) are transferred unaltered. Similarly, it is desired to transfer magnetic tapes files that have been generated with the Tektronix BASIC interpreter to disc files; they must be generated as ASCII files.

The goal of the example in *Figure 2* is to transfer the tape file 4 to the disc file CAT2. It is known that the information in file 4 is coded binary.

The goal of the example presented in *Figure 3* is to transfer file 4 on the tape to the type 4 file, CAT3, on the disc. The difference between the examples of *Figures 2* and *3* is that CAT3 did not exist; however, as indicated in *Figure 3*, it was created as the appropriate type file.

As with XFER, if TCOM encounters errors in accessing disc files, the program is automatically terminated and the file management negative error codes are displayed. In addition, if the magnetic tape unit on the 4051 detects errors, the appropriate message will be printed on the screen.

## C. BASIC TO DISC COMMUNICATIONS PROGRAM (TKHP)

The purpose of the BASIC to Disc Communications Program is to provide an interface between BASIC programs executing on the Tektronix 4051 and flexible discs residing in the HP-9885M disc drive. The program, TKHP, is written partly in FORTRAN and partly in assembler. A list of the program is given in Appendix E. The absolute binary version of the program is located on the FORTRAN DEV. DISC in the file TKHP.

It can be run under the control of the Type I systems on either the FORTRAN DEV. DISC or the ASSEMBLER DEV. DISC. The symbolic and relocatable binary files for the FORTRAN part of the program are located, respectively, in the files $TKHP and %TKHP, located on the FORTRAN DEV. DISC. The assembler part of the program has three entry points, HELP, INP, and OUT. The symbolic and binary relocatable versions of this part are located respectively, in the files $HELP and %HELP located on the ASSEMBLER DEV. DISC.

The principle of operation of TKHP is as follows. After starting the execution of the program, the user is asked to give the names of up to eight files that are to be made available for I/O with the Tektronix BASIC Interpreter. (At this point the user can also mount a different disc.) He can use any or all of the eight; however, he must remember which file number goes with which name. If any of the files do not exist, they will be created with lengths of 20 blocks (extents will be added if needed). The program tells the user which files are created, which files are not being used, and how many files are open. Then a "ready" indicator is transmitted to the screen of the 4051.

At this point the program is waiting for instructions from a BASIC program running on the 4051. These instructions must come in the form of ASCII character strings, followed by a carriage return. *Table 4* defines the instruction set, where n is a numerical from one to eight.

For a BASIC program to input a record of ASCII data from file ? on the disc, the following sequence can be used:

```
PRINT @40: "I?"(? Any file 1 through 8)
INPUT @ 40: (Variable list).
```

(Note: On input and output, variable lists cannot contain matrices; elements of matrices are acceptable.) To avoid the possibility of losing data there should be no statements separating these two. It is assumed that the variable list is compatible with the record that is forthcoming, i.e., the number of variables is equal to the number of numbers, etc. (A number has the normal definition for free field input with the BASIC INPUT statement.) By knowing the format of the data, the input can also be made under format control.

For the reading of matrix values (A) from disc to terminal, the following sequence should be used:

```
FOR I=1 to P
FOR J=1 to Q
PRINT @40: "I?" (? Any file 1
  through 8)
INPUT @40: A (I,J)
NEXT J
NEXT I
```

TABLE 4. COMMANDS FOR TKHP

| CHARACTER STRING | INTERPRETATION BY TKHP |
|---|---|
| Rn | Rewind file n . * |
| On | Output the following logical record to N. |
| In | Send the next logical record from N . |
| E | End execution of TKHP. |

** If n is omitted, a default to file 1 occurs.*

In order to output a record from a BASIC program to file "?," the following sequence must be used:

PRINT@40: "0?" (? any file 1 through 8)
INPUT@40: P$
PRINT@40: USING XXX: (variable list).

The first statements tell TKHP to prepare to receive a record of data. The second statement forces the BASIC program to wait until TKHP is prepared to receive the data (P$ can be any target variable. The character P is actually what is read in TKHP). The third statement outputs the record to the disc. In this statement XXX is used to denote the statement number of the format statement. Although a formatted output is not required, it is advisable in order to "pack" the data on the disc. Unformatted output can waste valuable disc space with blank characters. Outputting data to any of the other eight files should be obvious.

For each block of data to be read on the disc, the previous sequence must be repeated. Any number of variables can be printed from the terminal to the disc as long as the image statement reflects the number [e.g., IMAGE 4(3D)] of variables in the string (four in this case) and the variable list defines them (e.g., P, Q, R, S) with a carriage return after the last. Inherent in the TKHP program is a maximum number of 128 in any variable string.

For transferring a P by Q matrix A to the disc, the following sequence should be used:

```
FOR I=1 to P
FOR J=1 to Q
PRINT@40: "O?" (? any file 1 through 8)
INPUT@40: P$
PRINT@40: A(I,J)
NEXT J
NEXT I
```

Rewinding file ? using a BASIC program can be accomplished with the following statement:

PRINT @ 40: "R?".

If a file is being used for a scratch file, i.e., input and output in the same program, it should be rewound before inputting from it after output has occurred.

To illustrate the utility of TKHP in providing the link between the Tektronix BASIC and a flexible disc, the BASIC program shown in *Figure 4* was written. The program reads values for X, Y, and Z from file 1 and values for W and U from file 2. Five computations are made using X, Y, Z, W, and U, producing values for P, Q, R, V, and S. Then P, Q, R, V, and S are printed on file 3. This is repeated while reading two records from files 1 and 2. Then, file 1 is rewound, and the above is repeated except that records 1 and 2 of file 1 are used, respectively, with records 3 and 4 of file 2. A total of four records is printed on file 3. Finally, the BASIC program terminates the execution of TKHP and returns the 4051 to Terminal Mode.

The sequence of events that occur prior to and during the execution of the BASIC program in *Figure 4* is shown in *Figure 5*. TKHP is loaded into memory and run. The files TEST, TESTX, and TESTY are assigned the numbers one, two, and three, respectively. One or more blanks are entered for the other five files to indicate they are not being used. Then, the computer indicates that TESTY was not found on the disc; thus, it is created. The files not being used and the number of files open are printed. Then, the ready indication is received. The user then presses the return to BASIC key. The BASIC program in *Figure 4*, which had been previously loaded into memory of the 4051, is run. Upon completion of the BASIC program, the 4051 returns to Terminal Mode and the indication of the completion of TKHP is printed.

*Figure 6* shows the session with the HP-21 MX for aborting TKHP, loading the file manager (FMGR) and inspecting the files TEST, TESTX, and TESTY. Prior to execution of TKHP and the BASIC program, the files TEST and TESTX existed with the contents shown. However, the file TESTY was created and its contents were generated by the BASIC program.

## 4. SUMMARY AND CONCLUSIONS

In this report, software developments for fully utilizing the marriage between a Tektronix 4051 Graphic System, an HP-21 MX minicomputer, and an HP-9885M flexible disc drive have been presented. First, tables summarizing the software that has been specifically developed for this combination by this author were presented. Then, three programs that were developed as aids in data transfer between the devices were presented and discussed. The use of each of the programs was illustrated with one or more examples.

Software developments presented in this report have added to the flexibility of the

```
80 CALL "CMSET"
100 A$="I"
110 B$="O"
120 E$="E"
130 R$="R"
150 I1=1
160 I2=2
162 I3=3
165 FOR J=1 TO 2
170 FOR I=1 TO 2
180 PRINT @40:A$,I1
190 INPUT @40:X,Y,Z
192 PRINT @40:A$,I2
194 INPUT @40:W,U
200 P=X+Y
210 Q=X*Y+Z*W
220 R=X+Y+W+Z+U
230 V=2*X+3*W+4*U
240 S=X-Y+Z-W+U
250 PRINT @40:B$,I3
260 INPUT @40:P$
270 PRINT @40: USING 280:P,R,Q,S,V
280 IMAGE 5(10D)
290 NEXT I
292 PRINT @40:R$,I1
298 NEXT J
299 PRINT @40:E$
300 CALL "TERMIN"
310 END
```

Figure 4. BASIC program for showing utility of TKHP.

```
*RU,TKHP2
MOUNT DISC TO SEND AND/OR RECEIVE
GIVE NAME OF NO. 1 FILE
TEST
GIVE NAME OF NO. 2 FILE
TESTX
GIVE NAME OF NO. 3 FILE
TESTY
GIVE NAME OF NO. 4 FILE

GIVE NAME OF NO. 5 FILE

GIVE NAME OF NO. 6 FILE

GIVE NAME OF NO. 7 FILE

GIVE NAME OF NO. 8 FILE

FILE TESTY  NOT FOUND. IT IS CREATED
THERE IS NO NO. 4 FILE
THERE IS NO NO. 5 FILE
THERE IS NO NO. 6 FILE
THERE IS NO NO. 7 FILE
THERE IS NO NO. 8 FILE
 3 FILES ARE OPEN
* * * *  READY  * * * *

RUN
  TKHP2 : STOP    0000
```

Figure 5. Example execution of TKHP.

above combination of equipment. However, the addition of other equipment would enhance the flexibility even more. In particular, the addition of another flexible disc, another 32K of memory, of an HP-2644/45 data station and of a line printer would more than double the capability of the system.

```
*OF,TKHP ,8
TKHP  ABORTED

*LO,FMGR
 APLDR: DONE- FMGR

*ON,FMGR
:LI,TEST
TEST    T=00004 IS ON CR32760 USING 00001 BLKS R=0000

0001  1   4   5
0002  6   2   3

:LI,TESTX
TESTX   T=00004 IS ON CR32760 USING 00001 BLKS R=0000

0001  8  10
0002  1  7
0003  2  15
0004  10 25

:LI,TESTY
TESTY   T=00004 IS ON CR32760 USING 00020 BLKS R=0004

0001               5         28          44          4        66
0002               8         19          15         13        43
0003               5         27          14         15        68
0004               8         46          42         22       142

:
```

Figure 6.  Contents of files used to illustrate TKHP.

# APPENDIX A

The following is a listing of a BASIC program for setting certain environmental parameters and putting the Tektronix 4051 in Terminal Mode.

```
100 CALL “RATE“, 2400,5,0
110 CALL “MARGIN”, 0,0,0
120 CALL “TSTRIN”,@“J”, “D”
130 CALL “PROMPT”, 1,0,“R”
140 CALL “TERMN”
150 END
```

# APPENDIX B
# LISTING OF PROGRAM
# XFER

```
FTN,L,A
      PROGRAM XFER
      DIMENSION IDCB(144),IBUF(18000),NAME(3),LEN(999),ISIZE(2)
      IDCB(10)=0
      CALL DCMC(1,-2,0)
      WRITE(1,4)
4     FORMAT("DISMOUNT & MOUNT (IF DESIRED) AND GIVE FILE NAME")
      READ(1,5) (NAME(I),I=1,3)
5     FORMAT(3A2)
      CALL DCMC(0,2,0)
      CALL OPEN(IDCB,IERR,NAME,0,0,0,144)
      IF(IERR)12,14
12    WRITE(1,10)IERR
10    FORMAT("ERROR: FMP ",I3)
      STOP
14    I=1
      ITYPE=IERR
      ISIZE(2)=0
      J=1
      KTOT=0
15    CALL READF(IDCB,IERR,IBUF(I),128,LEN(J))
      IF(LEN(J))20,16
16    I=LEN(J)+1+KTOT
      IF(ITYPE-3)22,23
22    ISIZE(2)=LEN(J)
23    KTOT=KTOT+LEN(J)
      J=J+1
      IF(KTOT-18000)15,18
18    WRITE(1,19)
19    FORMAT("IBUF IS TOO SMALL")
      STOP
20    CONTINUE
      CALL CLOSE(IDCB)
      J=J-1
```

```
        ISIZE(1)=-1
        CALL DCMC(1,-2,0)
        WRITE(1,25)
25      FORMAT("REMOVE DISK & MOUNT NEW DISK & GIVE FILE NAME")
        READ(1,5) (NAME(I),I=1,3)
        CALL DCMC(0,2,0)
        CALL OPEN(IDCB,IERR,NAME,0,0,0,144)
        IF(IERR)31,40
31      WRITE(1,35)(NAME(I),I=1,3),ITYPE,ISIZE(1)
35      FORMAT("THE FILE ",3A2," IS CREATED.","  IT IS TYPE ",I2," AND "

       1   "IS ",I3," BLOCKS LONG.")
        CALL CREAT(IDCB,IERR,NAME,ISIZE,ITYPE,0,0,144)
40      CONTINUE
        I=1
        DO 50 K=1,J
        CALL WRITF(IDCB,IERR,IBUF(I),LEN(K))
        IF(IERR)43,45
43      WRITE(1,10) IERR
        STOP
45      I=LEN(K)+I
50      CONTINUE
        CALL LOCF(IDCB,IERR,IREC,IRB,IOFF,JSEC)
        ITRUN=JSEC/2-IRB-1
        CALL CLOSE(IDCB,IERR,ITRUN)
        WRITE(1,55)
55      FORMAT("TRANSFER COMPLETED")
        STOP
        END
        END$
EOF
```

# APPENDIX C
# EXAMPLE OF RUNNING PROGRAM XFER

```
*ON,XFER
DISMOUNT & MOUNT (IF DESIRED) AND GIVE FILE NAME
TEST
REMOVE DISK & MOUNT NEW DISK & GIVE FILE NAME
TESTX
THE FILE TESTX  IS CREATED.  IT IS TYPE  4 AND IS  -1 BLOCKS LONG.
TRANSFER COMPLETED
  XFER  : STOP    0000
```

# APPENDIX D
# LISTING OF PROGRAM TCOM

```
FTN,L,T
      PROGRAM TCOM
      DIMENSION IDCB(1296),IBUF1(128),IBUF2(258),NAME(3)
      COMMON N2,IBUF2,KSKIP
5     WRITE(1,10)
10    FORMAT("IF YOU ARE SENDING INFORMATION TO THE TAPE TYPE +1"
     1     /"IF YOU ARE SENDING INFORMATION FROM THE TAPE TYPE -1"
     2     /"IF YOU WANT TO TERMINATE THIS SESSION TYPE 0")
      READ(1,15) KX
15    FORMAT(I5)
      IF(KX)30,40,20
20    CALL TRTP(IDCB,IBUF1,IBUF2,NAME)
      GO TO 5
30    CALL TRDS(IDCB,IBUF1,IBUF2,NAME)
      GO TO 5
40    STOP
      END
      SUBROUTINE TRTP(IDCB,IBUF1,NAME)
      DIMENSION IDCB(1296),IBUF1(128),IBUF2(258),NAME(3)
      COMMON N2,IBUF2,KSKIP
      IDCB(10)=0
      CALL DCMC(1,-2,0)
      WRITE(1,4)
4     FORMAT("DISMOUNT & MOUNT AND GIVE FILE NAME")
      READ(1,5) (NAME(I),I=1,3)
5     FORMAT(3A2)
      CALL DCMC(0,2,0)
      CALL OPEN(IDCB,IERR,NAME,0,0,-2,1296)
      ITYPE=IERR
      IF(IERR)12,14
12    WRITE(1,13)IERR
13    FORMAT("ERROR: FMP",I3)
      STOP
14    K=0
```

```
      KOUNT=0
15    KOUNT=KOUNT+1
      CALL READF(IDCB,IERR,IBUF1,128,N)
      IF(N)80,16
16    IF(KOUNT-2)20,30
20    WRITE(1,25)ITYPE
25    FORMAT("THE FILE IS TYPE",I2,"    PREPARE THE TAPE"/
     1 "WHEN YOU ARE READY, TYPE -1 AND PRESS THE DATA RECEIVE KEY"

     2 /"TERMINAL CAN BE IN PROMPT MODE!")
      READ(1,27)KEY
27    FORMAT(I5)
      IF(KEY)30,20
30    CONTINUE
      IF(ITYPE-4)38,32,38
32    CONTINUE
      DO 35 I=1,N
35    IBUF2(I)=IBUF1(I)
      N2=N
      GO TO 51
38    CONTINUE
      K=0
      DO 50 I=1,N
      K=K+1
      NDUMB=IAND(IBUF1(I),37477B)
      IBUF2(K)=IOR(NDUMB,40100B)
      NDUMB=IAND(IBUF1(I),40300B)
      NDUMB=NDUMB/4
      K=K+1
      IBUF2(K)=IOR(NDUMB,40100B)
      IF(IBUF1(I))45,50
45    IBUF2(K)=IOR(IBUF2(K),1B)
50    CONTINUE
      N2=2*N
```

```
51      CONTINUE
        CALL OUTPT
        IF(N2)81,15
80      N2=N
        GO TO 51
81      CONTINUE
        DO 82 J=1,3
        K=0
        DO 82 I=1,20000
82      K=K+1
        CALL CLOSE(IDCB)
        WRITE(1,90)
90      FORMAT(/"TRANSFER TO TAPE COMPLETED")
        RETURN
        END
        SUBROUTINE TRDS(IDCB,IBUF1,NAME)
        DIMENSION IDCB(1296),IBUF1(128),IBUF2(258),NAME(3)
        COMMON K,IBUF2,KSKIP
        IDCB(10)=0
        KSKIP=1
        CALL DCMC(1,-2,0)
        WRITE(1,4)
4       FORMAT("MOUNT DISC TO RECEIVE; GIVE TYPE AND FILE NAME"
       1 /"FORMAT IS (I1,1X,3A2)")
        READ(1,5)ITYPE,(NAME(I),I=1,3)
5       FORMAT(I1,1X,3A2)
        CALL DCMC(0,2,0)
        CALL OPEN(IDCB,IERR,NAME,0,0,-2,1296)
        IF(IERR)12,14
12      WRITE(1,13) (NAME(I),I=1,3),ITYPE
13      FORMAT("FILE NOT FOUND. FILE, ",3A2," IS CREATED AS A TYPE:",I2,
       1 " .")
```

```
      CALL CREAT(IDCB,IERR,NAME,-1,ITYPE,0,0,1296)
14    CONTINUE
11    WRITE(1,15)
15    FORMAT("PREPARE TAPE: WHEN READY TYPE -1, PRESS RETURN "
     1   " AND PRESS DATA SEND KEY"/"TERMINAL MUST BE IN PROMPT MODE")
      READ(1,99)IDOG
99    FORMAT(I5)
      IF(IDOG)20,11
20    K=0
      CALL INSUB
      IF(K)80,25
25    CONTINUE
      IF(ITYPE-4)39,35,39
35    K2=K/2
      DO 37 I=1,K2
37    IBUF1(I)=IBUF2(I)
      GO TO 51
39    K2=K/4
      K=0
      DO 50 I=1,K2
      K=K+1
      IY= IAND(IBUF2(K),37477B)
      K=K+1
      IZ= IAND(IBUF2(K),10060B)
      IZ= IZ * 4
      IBUF2(K)= IAND(IBUF2(K),1B)
      IF(IBUF2(K))41,42,41
41    IZ= IOR(IZ,100000B)
42    IBUF1(I)= IOR(IY,IZ)
50    CONTINUE
51    CALL WRITF(IDCB,IERR,IBUF1,K2)
      IF(IERR)55,20
55    WRITE(1,56) IERR
56    FORMAT("ERROR: FMP-",I3)
```

```
        GO TO 90
80      CONTINUE
        DO 82 J=1,3
        K=0
        DO 82 I=1,20000
82      K=K+1
        WRITE(1,85)
85      FORMAT(/"TRANSFER COMPLETED")
        CALL LOCF(IDCB,IERR,IREC,IRB,IOFF,JSEC)
        ITRUN=JSEC/2-IRB-1
90      CALL CLOSE(IDCB,IERR,ITRUN)
        RETURN
        END
        END$
```

```
ASMB,L,T
       NAM OUTPT,7
       ENT OUTPT
       EXT IO13
       COM N2,IBUF2(258)
SC     EQU 13B
OUTPT  NOP
       LDA N2
       ALS
       CAX
       LIA SC
       SSA
       JMP *-2
       CLF 00
       LDA CNW2
       OTA SC
       LDA DC2
       JSB IO13
       CXA
       SSA
       JMP CFILE
       LDB ADR2
UP1    LIA SC
       SSA
       JMP *-2
       LDA CNW2
       OTA SC
       LBT
       JSB IO13
       DSX
       JMP UP1
       LIA SC
       SSA
       JMP *-2
```

```
       LDA CNW2
       OTA SC
       LDA DC4
       JSB IO13
       CLF SC
       STF 00
       JMP OUTPT,I
CFILE  LIA SC
       SSA
       JMP *-2
       LDA CNW2
       OTA SC
       LDA EOT
       JSB IO13
       CLF SC
       STF 00
       JMP OUTPT,I
ADR2   DBL IBUF2
DC2    OCT 22
DC4    OCT 24
EOT    OCT 4
CNW2   OCT 120000
       END
EOF
```

```
ASMB,L,T
       NAM INSUB,7
       ENT INSUB
       EXT IO13
       COM K,IBUF2(259),KSKIP
SC     EQU 13B
INSUB  NOP
       LIA SC      LOAD I.F.C. REG.
       SSA                TEST BUSY BIT
       JMP *-2            IF ZERO, SKIP THIS ONE
       CLF 00             OF INTERRUPT
       LDA KSKIP
       SSA
       JMP DXN1
       LDA NEG1
       STA KSKIP
       LDA CNW1
       OTA SC
       JSB IO13
       LIA SC
DXN1   NOP
       LDA CNW2           LOAD A WITH OUT-WORD
       OTA SC             SEND TO I.F.C.
       LDA DC2            LOAD ROPEN CHAR.
       JSB IO13           SEND TO TEK. TERM.
       LDA CNW2
       OTA SC
       LDA DC2            LOAD A WITH PROMPT CHAR.
       JSB IO13
       LDA CNW2
       OTA SC
       LDA DC4            LOAD A WITH RCLOSE CHAR.
       JSB IO13
       LDX =D0            LOAD X WITH ZERO
```

```
         LDB ADR1            LOAD B WITH BYTE ADDR. OF IBUF2
RET1     LDA CNW1            LOAD A WITH IN-WORD
         JSB IO13
         LIA SC
         CPA NUL             IS CHAR. TOPEN?
         JMP UP1             YES! JUMP.
         CPA DC3             NO! IS IT TCLOSE?
         JMP DWN3            YES! JUMP.
         JMP DWN1
UP1      LDA CNW1
         JSB IO13
         LIA SC
         CPA EOT
         JMP DWN2
DWN1     CPA LF
         JMP RET1
         ISX                 NO! COUNT CHAR.
         SBT                 STORE CHAR. IN BYTE OF IBUF2.
         JMP RET1            JUMP AND READ ANOTHER.
DWN2     LDA =D-1            SET END OF FILE FLAG.
         STA K
         JMP DWN4
DWN3     STX K               STORE CHAR. COUNT IN K
DWN4     CLF SC
         STF 00              ON INTERRUPT
         JMP INSUB,I         RETURN
ADR1     DBL IBUF2           DFN. BYTE ADDR. OF IBUF2
CNW1     OCT 140000          INPUT WORD
CNW2     OCT 120000          OUTPUT WORD
NUL      OCT 0               CTRL
DC2      OCT 22              CTRL R
DC3      OCT 15              CTRL M
DC4      OCT 24              CTRL T
EOT      OCT 4               CTRL D
```

```
LF      OCT 12          LINE FEED
NEG1    DEC -1
        END
EOF
```

# APPENDIX E
# LISTING OF PROGRAM TKHP

```
FTN,L,T
      PROGRAM TKHP2
      DIMENSION IDCB(528,8),IBUF(128),NAME(3,8),KD(8)
      COMMON KSKIP,K,L,IBUF
      DO 1 I=1,8
      IDCB(10,I)=0
1     KD(I)=0
      CALL DCMC(1,-2,0)
      WRITE(1,2)
2     FORMAT("MOUNT DISC TO SEND AND/OR RECEIVE")
      DO 4 J=1,8
      WRITE(1,3) J
3     FORMAT("GIVE NAME OF NO. ",I1," FILE")
4     READ(1,5) (NAME(I,J),I=1,3)
5     FORMAT(3A2)
      CALL DCMC(0,2,0)
      DO 438 J=1,8
      IF(NAME(1,J)-40B)430,432,430
430   IF(NAME(1,J)-20040B)434,432,434
432   WRITE(1,433) J
433   FORMAT("THERE IS NO NO. ",I1," FILE")
      KD(J)=-1
      GO TO 438
434   CALL OPEN(IDCB(1,J),IERR,NAME(1,J),0,0,-2,528)
      IF(IERR)455,438
455   WRITE(1,456) (NAME(I,J),I=1,3)
456   FORMAT("FILE ",3A2," NOT FOUND. IT IS CREATED.")
      CALL CREAT(IDCB(1,J),IERR,NAME(1,J),20,4,0,0,528)
      IF(IERR)40,438
438   CONTINUE
      KIN=8
      DO 482 I=1,8
482   KIN=KIN + KD(I)
      WRITE(1,484)KIN
```

```
484   FORMAT(I2," FILES ARE OPEN")
23    WRITE(1,24)
24    FORMAT("* * * *  READY  * * * *")
      CALL DELAY(4,20000)
      KOK=0
25    L=1
      CALL HELP
      IF(KOK)33,34
33    N=K/2
      CALL WRITF(IDCB(1,LO),IERR,IBUF,N)
      IF(IERR)40,35
35    KOK=0
34    IF(KSKIP)40,36,29
29    IF(KSKIP-2)32,31
31    CALL RWNDF(IDCB(1,L))
      GO TO 25
32    CALL READF(IDCB(1,L),IERR,IBUF,128,N)
      IF(N)40,28
28    K=2*N
      CALL OUT
      GO TO 25
36    CONTINUE
      KOK=-1
      LO=L
      CALL INP
      GO TO 25
40    CONTINUE
      DO 50 J=1,8
      IF(KD(J))50,42
42    CALL CLOSE(IDCB(1,J))
50    CONTINUE
      STOP
      END
      SUBROUTINE DELAY(M,N)
```

```
      DO 10 I=1,M
      L=0
      DO 10 J=1,N
10    L=L+1
      RETURN
      END
      END$
EOF
```

```
ASMB,L,T
          NAM HELP,7
          COM KSKIP,N2,L,IBUF1(128)
          ENT HELP,OUT,INP
          EXT IO13
ONE       OCT 1
ZERO      DEC 0
MINUS     DEC -1
TWO       DEC 50
THREE     DEC 51
FOUR      DEC 52
FIVE      DEC 53
SIX       DEC 54
SEVEN     DEC 55
EIGHT     DEC 56
RWN       OCT 122
OUTC      OCT 117
ENDC      OCT 105
PCH       OCT 120
CR        OCT 15
INW       OCT 111
CNW       OCT 140000
SC        EQU 13B
CNW2      OCT 120000
ADR       DBL IBUF1
HELP      NOP
          LIA SC
          SSA
          JMP *-2
          CLF 00
UP1       LDA CNW
          JSB IO13
          LIA SC
          CPA INW
```

```
          JMP DWN1
          CPA OUTC
          JMP DWN2
          CPA RWN
          JMP DWN7
          CPA ENDC
          JMP DWN3
          JMP UP1
DWN1      LDA ONE
          STA KSKIP
          JMP DWN4
DWN2      LDA ZERO
          STA KSKIP
          JMP DWN4
DWN7      LDA TWO
          STA KSKIP
          JMP DWN4
DWN3      LDA MINUS
          STA KSKIP
DWN4      LDA CNW
          JSB IO13
          LIA SC
          LDB ONE
          ADB =D1
          CPA TWO
          STB L
          ADB =D1
          CPA THREE
          STB L
          ADB =D1
          CPA FOUR
          STB L
          ADB =D1
          CPA FIVE
```

```
      STB L
      ADB =D1
      CPA SIX
      STB L
      ADB =D1
      CPA SEVEN
      STB L
      ADB =D1
      CPA EIGHT
      STB L
      CPA CR
      JMP DWN6
      JMP DWN4
DWN6  CLF SC
      STF 00
      JMP HELP,I
OUT   NOP
      LDB ADR
      LDX N2
      LIA SC
      SSA
      JMP *-2
      CLF 00
UP2   LDA CNW2
      OTA SC
      LBT
      JSB IO13
      DSX
      JMP UP2
      LDA CNW2
      OTA SC
      LDA CR
      JSB IO13
      CLF SC
      STF 00
      JMP OUT,I
INP   NOP
      LIA SC
      SSA
      JMP *-2
      CLF 00
      LDA CNW2
      OTA SC
      LDA PCH
      JSB IO13
      LDA CNW2
      OTA SC
      LDA CR
      JSB IO13
      LDB ADR
      LDX ZERO
UP3   LDA CNW
      JSB IO13
      LIA SC
      CPA CR
      JMP DWN5
      ISX
      SBT
      JMP UP3
DWN5  CXA
      CPA ZERO
      JMP UP3
      LDA ZERO
      SBT
      ISX
      STX N2
      CLF SC
      STF 00
      JMP INP,I
      END
EOF
```

## APPENDIX F
## LISTING OF FILE TYPES

| | |
|---|---|
| 0 | non-flexible disc file |
| 1 | fixed length 128-word record |
| 2 | fixed length records; user defines length |
| 3 | variable length record, sequential access, automatic extents |
| 4 | ASCII code and source programs (otherwise like type 3 files) |
| 5 | relocatable binary code (otherwise like type 3 files) |
| 7 | absolute binary (otherwise like type 3 files) |

## REFERENCES

1. *Data Communications Interface Manual*, Tektronix, Inc., Beaverton, Oregon, 1976.
2. *RTE-M Programmer's Reference Manual*, Hewlett-Packard, Cupertino, California, 1977.

## LIST OF ABBREVIATIONS AND SYMBOLS

| | |
|---|---|
| ASSM | ASSEMBLER |
| @ | Control @ . Obtain by pressing control and @simultaneously |
| D | Control D character. Obtained by pressing control and D character simultaneously |
| FMP | File Management Package |
| FORT. | FORTRAN |
| J | Control J. Obtained by pressing control and J simultaneously |
| lu | Logical unit number |
| PROG. | PROGRAM |
| R | Control R. Obtained by pressing control and R simultaneously |
| TCOM | Tape Communications Program |
| TKHP | BASIC to Disc Communications Program |
| XFER | Program for transferring files from one disc to disc |
| 4051 | Tektronix 4051 graphic system |
| 9885M | HP-9885M flexible disc drive |

# DISTRIBUTION

| | No. of Copies |
|---|---|
| Defense Documentation Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 12 |
| IIT Research Institute<br>ATTN: GACIAC<br>10 West 35th Street<br>Chicago, Illinois 60616 | 1 |

| | | No. of Copies |
|---|---|---|
| DRSMI-LP, | Mr. Voigt | 1 |
| DRDMI-TG, | Mr. Huff | 1 |
| DRDMI-T, | Dr. Kobler | 1 |
| DRDMI-TGC, | Mr. Griffith | 40 |
| DRDMI-TBD | | 3 |
| DRDMI-TB | | 3 |
| DRDMI-TI | (Record Set) | 1 |
| | (Reference Copy) | 1 |